Autonomous Event Management Systems
based on Software Agents

Lorena Bearzotti*
lbearzot@ceride.gov.ar

Enrique Salomone*
salomone@ceride.gov.ar

Omar Chiotti*
chiotti@ceride.gov.ar

*INGAR – CONICET, Avellaneda 3657, Santa Fe, Argentina, 3000
GIDSATD- UTN FRSF, Lavaisse 610, Santa Fe, Argentina, 3000

Abstract
Event management is a complex problem that begins with the capture of any relevant unexpected event, followed by an analysis of this event to assess whether it produces an exception in the execution leading to a present or future deviation from the plan. If an exception is detected, the decision-maker has to evaluate if the safety margins of the plan can mitigate the effect of this exception and then to define the control actions to be carried out. If the effect of this exception cannot be satisfactorily mitigated, the decision-maker has to re-plan the process.

In this work we describe an approach to systematically support the management of events during the execution of a plan. Based on this approach we are developing a framework that can be used to implement software applications to support the event management for specific domains. We describe the events management problem, our approach, an application example and the architecture of a multi-agent system designed to support the supply chain events management.

Keywords: Software Agents, event management, support system

Resumen
La gestión de eventos es un problema complejo que comienza con la captura de algunos eventos inesperados relevantes, seguido por un análisis de este evento para comprobar si el mismo produce una excepción en la ejecución presente o futura del plan. Si una excepción es detectada, el tomador de decisión tiene que evaluar si los márgenes de seguridad del plan pueden mitigar el efecto de esta excepción y entonces definir las acciones de control a realizarse. Si el efecto de esta excepción no puede mitigado satisfactoriamente, el tomador de decisión debe replanificar el proceso.

En este trabajo describimos una propuesta para dar soporte a la gestión de eventos durante la ejecución de un plan. Basándonos en esta propuesta hemos desarrollado un framework que puede ser usado para implementar software de aplicación para soportar la gestión de eventos para un dominio específico. Describimos el problema de gestión de eventos, nuestra propuesta, un ejemplo de aplicación y la arquitectura de un sistema multiagente diseñado para soportar la gestión de eventos en la cadena de suministro.

Palabras Claves: Agentes de Software, gestión de eventos , sistemas soporte
Introduction

A main functionality of many information systems is to support the planning activity. Once a plan has been defined, the next activity is to monitor its execution with the aim of detecting exceptions that require corrective actions. These exceptions are produced by unexpected events. Then, it is necessary to monitor the events to detect exceptions of a plan that is being executed, and to carry out actions to adjust it to the current context. This process is called event management.

The occurrence of unexpected events is a natural characteristic of real world and it is a well-known challenge for the planning activity. It is widely accepted that a robust plan has to be defined with certain flexibility for adapting to new conditions when it is executed. This is why decision-makers normally define their plans with "slacks" to confront variability. That is, each plan includes safety margins that can be used to mitigate the effect of exceptions.

In this work we describe an approach to systematically support the management of events during the execution of a plan. Based on this approach we are developing a framework that can be used to implement software applications to support the event management for specific domains. This paper is organized as follow. In section 2 we describe the events management problem. In section 3 we describe our approach. In section 4 we present an application example and the architecture of a multi-agent system designed to support the supply chain events management.

1. The Event Management Problem

Event management is a complex problem that begins with the capture of any relevant unexpected event, followed by an analysis of this event to assess whether it produces an exception in the execution leading to a present or future deviation from the plan. If an exception is detected, the decision-maker has to evaluate if the safety margins of the plan can mitigate the effect of this exception and then to define the control actions to be carried out. If the effect of this exception cannot be satisfactory mitigated, the decision-maker has to re-plan the process. The activities involved in an event management process are semi-structured decision-making processes (Simons 1987). That is, some parts of these decision processes can be automated, but other parts require the human judgment and then they cannot be automated.

An event management system (EMS) can be implemented with different levels of automation. The simplest level consists in a manual monitoring of the process execution. When the decision-maker detects an exception, she adopts a convenient control action or notifies this exception to the planning level for re-planning decisions.

A higher level of automation is the use an alarm system. An alarm system automates the monitoring of the process execution to detect an event that could produce an exception and notifies it to the decision-maker. However, a typical alarm system does not assess the severity of the exception nor it defines control actions to implement. Therefore, these two activities have to be executed by the decision-maker. That is, an alarm system will be permanently notifying events to the decision-maker without analyzing their severity, and without taking into account the information about safety margins of a plan that can be used to mitigate the effect of these exceptions.

A more efficient EMS can be envisioned by giving the system some autonomy to define control actions to mitigate the effects of exceptions. These control actions can be defined within the safety margins specified in the plan. If the planning system sends to the EMS a plan without safety margins, EMS will turn on an alarm system since it has not margins of action. A plan without safety margins is inflexible, cannot be adapted to changes and will require frequent re-planning.

The event management problem is naturally a control problem. Then, it is possible to identify the three following types of variables.

**Observed variable:** This variable is observed during the process execution with the purpose of detecting the occurrence of an unexpected event. This event is triggered by a change in the value of the observed variable.

**Controlled variable or State variable:** This variable defines the control point. It has both a plan of states and the safety margins defined by the planning system. Then, an exception is the gap between the current and the planned values of this variable. A plan of states is a succession of values that the state variable has to take on along the planned time horizon. It is represented as a list of n-uple \([e,t]\) where \(e\) is the planned value that the state variable has to take on time \(t\).

**Decision variable:** it is the independent variable whose value can be adjusted to mitigate the effects of an exception with the purpose of bringing the system back to the specified objectives. The value change of this variable affects the value of both the controlled variable and the monitored variable.

In Figure 1 we represent the state variable as a time function. We highlight a point \([e,t]\), that represents the n-uple \([e,t]\) of the state plan. It represents a planned value \(e\) that the state variable has to take on time \(t\). We also represent the slack of
the state plan associated to this point \([e, t]\) through the safety margins \(\Delta e\) and \(\Delta t\). \(\Delta e \times \Delta t\) defines the set of values that the n-uple \([e, t]\) could take without making infeasible this plan. This defines the case represented as scenario 1 in the Figure 1.

A second case defined as scenario 2 in the Figure 1, represent a value of the n-uple \([e, t]\) that turn on infeasible this plan.

![Figure 1: Scenarios of a state variable](image)

2. Event Management System

In this section we present an approach to develop an automated EMS. Such a system should be able to monitor events to detect exceptions and to support the process for analyzing the safety margins of a plan to define control actions to mitigate the effect of these exceptions. When these control actions cannot be automatically inferred, EMS may interact with the decision-maker to define them. Furthermore, if there are no appropriate control actions within the margins of the plan, EMS has to notify the exception to the planning level for re-planning decisions.

Event management is a complex control problem. For any practical application it involves thousands of checkpoints. The affected variables are usually interrelated and these relationships may not be explicit in a global perspective. Our approach is based on the decomposition of the global problem into a set of simpler interwoven sub-problems. The distribution is performed through out the state variables defining for each of them a control point that is represented by a subsystem. Each subsystem has also defined its observed variables and its decision variables. Then, the EMS can be modeled as a network of control points that represents the relationship among their state variables. In this way, the causal relationship among state variables can be explicitly defined pair wise and the global interactions described by the network.

The objectives of the set of subsystems are defined with the aim that the EMS can reach its objective. Each subsystem has to perform a state plan, derived from the EMS global plan. The objective of each subsystem is to control its own variables according to the state plan. This requires a collaborative relationship among the subsystems that define the EMS.

A planning system (PS) defines the plan for a time horizon. The EMS receives this plan including the corresponding safety margins. Also, the PS is responsible for re-planning whenever when an exception cannot be controlled by the EMS, yielding the current execution infeasible. In Figure 2, we represent this relationship between the PS and the EMS.
The EMS is modeled as a set of interrelated sub-systems, each of them representing a control point that has a plan and its safety margins defined by the PS. Each subsystem has to monitor its observed variables to detect the occurrence of an event and to analyze it to detect whether it produces an exception. When an exception is detected, the subsystem has to support the process for analyzing the safety margins of the plan to define control actions to mitigate the effect of this exception. If appropriate control actions cannot be implemented within the limits of the local safety margins, the subsystem has to negotiate with its related subsystems to reach a collaborative control action to mitigate the effect of this exception. If appropriate joint control actions cannot be defined, the subsystem has to notify the exception to the planning level for re-planning decisions.

The unexpected events that can affect a subsystem can be classified in three types:

- **Internal event**: It is an unexpected event detected by a subsystem monitoring its observed variable. This type of event can produce an internal exception of the subsystem, and when it cannot be mitigated through internal control actions neither through control actions collaboratively defined by the related subsystems it raises an EMS exception.

- **External event**: It arrives to the subsystem from other related control point (sub-system) that requires collaboration to mitigate an exception. This event generates an exception in the subsystem named external exception.

- **Simulated event**: It is an event generated by the subsystem itself with the purpose of to evaluate futures scenarios to select a solution for the current situation. An exception that can be generated by this event type is named simulated exception.

The described model is strongly based in autonomy of the subsystems. This is a typical characteristic of software agent technology [12][4][5]. Therefore, the EMS entities can be modeled by autonomous or semi-autonomous agents where the knowledge and abilities are locally distributed and the communication between them is performed through messages [8]. Then, from the agent vision, the proposed model for the EMS is an agency [1] that defines a machine developed to support the event management by means of the cooperation among agents.

### 2.1 Event Management Process

To manage the events we propose a process that can to manage two possible scenarios:

**Scenario 1**: This case occurs when the state variable represented by the n-uple \( e,t \) belongs to the \( \Delta e \times \Delta t \) area defined by the safety margins of the plan (as represented in Figure 1). When this happens, the subsystem makes a simulation to analyze the future implications of the variation of the state value with the purpose of determining if at some future time an exception could occur. If it so, the subsystem has two alternatives: to mitigate this future exception with its safety margins, or it can initiate a negotiation with its related subsystems to jointly define a control action to mitigate this future exception. The objective of the negotiation is to distribute the exception among related subsystems. If appropriate joint control actions cannot be defined, the sub-system has to define that an EMS exception will occur and then to notify it.

**Scenario 2**: This case occurs when the state variable represented by the n-uple \( e,t \) falls outside the \( \Delta e \times \Delta t \) area defined by the safety margins of the plan (as represented in Figure 1). In this case an exception for the subsystem has occurred. The subsystem cannot mitigate it, and then it has to initiate a negotiation with its related subsystems to jointly define a
control action to mitigate this exception. If appropriate jointly control actions cannot be defined, the subsystem has to define that an EMS exception has occurred and then to notify it. Contrarily, if jointly control actions can be used to mitigate this exception, this allows the definition of a collaborative solution to be implemented by all the related subsystems that contributes to reach this solution.

We represent this process through a pseudo-code algorithm:

Read the value of the observed variable

\[ \text{IF observed variable has variation THEN} \]

Test the impact in the state variable

\[ \text{CASE scenario for the state variable} \]

1. \textbf{Scenario 1:} carry out a simulation

\[ \text{IF exception occur THEN} \]

Start negotiation process

\[ \text{IF negotiation process THEN} \]

Implement solution

\[ \text{ELSE} \]

Absorb variation with the safety margins

\[ \text{ENDIF} \]

\[ \text{ENDIF} \]

2. \textbf{Scenario 2:} The event defines an exception

Start negotiation process

\[ \text{IF negotiation process THEN} \]

Implement solution

\[ \text{ELSE} \]

Notify the current exception

\[ \text{ENDIF} \]

\[ \text{END CASE} \]

\[ \text{ENDIF} \]

2.2 Example of application: Supply Chain Event Management

EMS is a helpful abstraction for deriving specialized frameworks. An example of this specialization is the event management in the domain of supply chains.

A Supply Chain (SC) can be defined as a network of autonomous or semi-autonomous business entities, collectively responsible for procurement, production and distribution activities associated with one or more families of related products. [11]

Modern supply chains operate in a highly dynamic environment. Global markets, customer oriented processes, shorter product life cycles and increasing competition bring constant changes to the supply chain. These changes unavoidably drive to more complex, uncertain and variable operations.

Evolution of supply chains also drives the evolution of their management. Supply Chain Management (SCM) has emerged in the organizational landscape as a new way to conduct business. SCM is a set of approaches utilized to efficiently integrate suppliers, manufacturers, warehouses, and stores, so that merchandise is produced and distributed at the right quantities, and at the right time, in order to minimize system wide costs while satisfying service level requirements [10]

The management of events during the execution of supply chain operations becomes a critical task for business success. New information technology is key to perform effective supply chain event management (SCEM) [2][3][7][9][6]. In this context, SCEM is defined as the business process where significant events are timely recognized, reactive actions are quickly triggered, the material and information flows are adjusted based on business rules and the notification of key employees is immediate.

SCEM is a component in SCM that bridges the gap between the planning and execution processes. In the domain of supply chains, a plan is defined as the replenishment of materials in different locations, represented by production,
packaging and transfer orders. The execution of these orders is subject to unexpected variability and the objective of SCEM is to react against this variability minimizing the deviation from what was planned.

The typical perturbing events in the supply chain domain can be classified as follows:

- **Boundary events**. Represent a change either in the time or in the quantity of any planned Input/Output order. For example, a cancellation of a customer order.

- **Resource unavailability**. Represent any unexpected change on the availability of the resources affected to the execution of an operation. For example, any equipment breakdown.

- **Process extension shortfall**. Mainly related to the intrinsic variability of processes where the actual amount obtained is less than expected or is delayed. For example, quality issues reduce the expected amount of a product.

The approach described for generic EMS proposed to model the system as network of control points. In the SC context, we propose to this network to be composed of *inventory control points*. These inventory control points are interconnected by *supply processes* that require *resources* to be executed. Given the nature of the events that can occur during the execution of supply chain operations, besides inventory, both supply processes and resources also need to have associated control points.

### 3. Design of the Architecture for a SCEM information system

We have modeled the SCEM system as a network of inventory control points that are linked among them by supply process control points. The supply processes use resources that also are modeled as resource control points. Then, in this model we define three types of control points:

- **SKU (Storage Keeping Unit)**: for the management of events associated to the inventory. It is defined by three main attributes: material, packing and location. It is represented by an n-uple \([m, p, l]\), where \(m\) is the material, \(p\) is the packing and \(l\) is the location.

- **RKU (Resource Keeping Unit)**: for the management of events associated to a resource. There will be a RKU for each resource of the supply chain. Its attributes define the resource availability that it represents.

- **SP (Supply Process)**: for the management of events associated with a supply process. A supply process represents the transition from one or more SKUs to another or other SKUs. There are three basic transitions: material change \((\Delta m)\), packing change \((\Delta p)\) and location change \((\Delta l)\). A supply process can involve one of these transitions or a combination of them. For example, the translation of a material \(m_1\) in packing \(p_1\) from the warehouse \(w_1\) to the warehouse \(w_2\) is represented by the supply process control point \(SP[\Delta l]\) that links the inventory control point \(SKU[m_1, p_1, w_1]\) with the inventory control point \(SKU[m_1, p_1, w_2]\).

To develop a multi-agent system (MAS) to support the SCEM we have used the Gaia technology [14] to analyze and design it.

Gaia methodology proposes a developer to think of building agent-based systems as a process of organizational design. The main concepts in Gaia are divided into two categories: abstract and concrete. The abstract concepts involve the **Roles**. A role is defined by four attributes: Protocols, Permissions, Activities and Responsibilities. Responsibilities are divided into two types: Liveness Properties and Safety Properties. The concrete concepts involve: Agent types, Services and Acquaintances. A role can be viewed as an abstract description of an entity’s expected function. A protocol defines the way that a role can interact with another role. The analysis stage implies defining the role model and the interaction (protocols) model. In this work, the terms of Gaia are defined in the different sections where they are necessary. More details of Gaia methodology can be obtained in [Wooldridge, et al, 2000]

This methodology supports the definition of a detailed design of the MAS architecture and its components. However, this methodology does not provide a procedure to specify roles and protocols [15][16]. To extend the Gaia methodology to support roles and protocols specification, we used the approach of Villarreal, et al, (2002) which consists of three stages:

**Stage 1.** Define the information system’s goal:

“Minimize the disturbances of a plan that can be caused by unexpected events using the safety margins associated to this plan”.

---

PDF created with FinePrint pdfFactory Pro trial version [http://www.fineprint.com](http://www.fineprint.com)
Stage 2. Detailed definition and description of processes needed to achieve that goal.

The information system is modeled as a set of processes, defining for each of them its inputs and outputs, and the precedence relationships among the activities involved in them. In SCEM we identified the *event management* process as the main process, and two associated sub processes: *event monitoring* process and *exception control* process. We represent the inputs and outputs of these processes in Figure 3.

![Diagram of SCEM processes](image)

Figure 3: inputs and outputs of the SCEM processes

The inputs and outputs of the defined sub processes are the following:

*Plan:* succession of values defined by the PS that the state variables have to take on along the planned time horizon.

*Safety margins* represent the slack of the plan. They define the set of values that the state variables could take without making the plan infeasible.

*Events:* facts from the environment that can impact the plan.

*Exceptions:* deviations of the process execution from its plan.

*System Exception:* deviation of the process execution that makes infeasible the current plan.

*Control actions:* adjusts of the decision variables value to mitigate the effects of an exception with the purpose of bringing the system back to the specified objectives.

In Figure 4 (a) and (b) we show the activities diagrams for the *event monitoring* and the *exception control* sub processes respectively.
Figure 4: activities diagrams; (a) of the event monitoring sub process; (b) of the exception control sub process

To complete the processes definition, the activities attributes are defined. Each of the defined process activity is described in detail. As an example, Figure 5 presents the AnalyzingEvent activity, which correspond to the event monitoring sub process.

<table>
<thead>
<tr>
<th>Activity: Analyzing Event</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective:</strong> Given the plan and the safety margins, evaluate if the event affects the state variable producing an exception</td>
</tr>
<tr>
<td><strong>Inputs:</strong> State Plan, Event</td>
</tr>
<tr>
<td><strong>Outputs:</strong> Exception</td>
</tr>
<tr>
<td><strong>Restrictions:</strong>--</td>
</tr>
<tr>
<td><strong>Resources:</strong>--</td>
</tr>
</tbody>
</table>

Figure 5: The Analyzing Event activity

**Stage 3.** Identify Roles and Protocols.

In this stage we follow the Gaia methodology. By grouping related activities of the defined processes we can identify the roles and the protocols. Once the roles and protocols are identified, we do detailed description of each role and each protocol. This defines the roles model. As an example, Figure 6(a) shows a schema of the InventoryMonitor role and Figure 6(b) shows a schema of the InventoryController role.
The defined roles have to be mapped to agents. We have defined three types of agents: RKU agents, SKU agents and SP agents. This defines the agent’s model. We represent this mapping on Figure 7, where we show the assignment a role to an agent. SKU agents perform the `InventoryMonitor` and the `InventoryController` roles. RKU agents perform the `ResourceMonitor` and the `ResourceController` roles; and SP agents perform the `SupplyProcessMonitor` and the `SupplyProcessController` roles.

Finally, the architecture of the multi-agent system is defined, where the Acquaintance model is specified. This is represented in Figure 8.

**Figure 6:** (a) schema of the `InventoryMonitor` role - (b) schema of the `InventoryController` role

**Figure 7:** agents model for the SCEM multi-agent system

**Figure 8:** SCEM multi-agent system architecture
3.1 SCEM Agents description

In this section we describe the main agents of the SCEM agency. These are the one with specific tasks related to the management of events in the supply chain. In addition to these agents, other types of agents, which are nominated as service agents, also compose the SCEM agency. Among the functions they perform we can mention the interfacing with users and other systems such as: ERP system, Plant Information system, Production Execution system, Transport Management system and Planning system.

SKU Agent

SKU agents represent a control point where the management of events related to inventory is done. Its three basic attributes are Product, Package and Location. By product we refer to every material resource, including raw materials, finished goods and in-process materials. Its function is to monitor the inventory to detect variations between what is planned and what is actually happening as the plan is executed.

An SKU Agent has an Input/Output List that determines to which SP agents it is related and if the relationship is input or output. Every item in the list has the following structure:

[I/O, Supply Process, Start Time, Rate, Duration]

This list is received from the Planning System through the corresponding interface agent. The SKU agent also receives the safety margins associated to each input/output activity represented as slacks both in quantities or time. From the Input/Output List, the SKU agent can create its state plan by computing the successive inventory states along the planning horizon. The state variable for the SKU agent is the inventory.

The relevant events are detected by comparing the current value with the planned state. The current value is obtained through the corresponding interface agent.

The monitored variable for the SKU is the Current Inventory value. The controlled variables are parameters of the Input/Output List.

The SKU Agent manages the following types of events:

Internal: related to a direct variation of the inventory variable. For example this event will be caused by inventory damage, expiration, accounting updates, etc.

External: variations in time or quantities in the Input/Output list that are originated by an SP agent. The variation is notified to the SKU agent by messages emitted by the SP agent.

Simulated: A simulated event is originated as part of the solution process and represents what-if situations.

These events may produce exceptions that will trigger a negotiation process. This negotiation has the objective of distributing the effect of a variation for being absorbed or minimized with the margins defined at each point avoiding a re-planning. For the SKU agent, the negotiation is performed through a dialog with SP agents that are related by input/output variations.

The details of the negotiation process are beyond the scope of this work and not described here.

RKU Agent

RKU Agents represent control points for the management of events related to resources other than material resources. There is one agent for each relevant resource in the supply chain. This agent monitors the availability of the corresponding resource. Examples of these resources are: Process Units, Transport equipment, Warehouses, etc.

For the RKU agent, the concept of plan is represented by a Usage Agenda. From this agenda it is possible to create a load profile that is the state plan for the resource. Every item in the Usage Agenda has the following structure:

[Supply Process, Start Time, Duration, Required Capacity]

An RKU agent only exchanges messages with SP agents. The usage agenda also track which SP agents are booking the resource during the planning horizon.

The state variable for the RKU agent is its availability state. The control variable is the timing in the usage agenda, either to postpone or expedite requirements. It is important to note that this control does not include re-scheduling of the requirement (a task that belongs to the Planning system). The control is only allowed within the idle windows.
representing the safety margins defined by the plan. Monitored variables depend on the specific resource and are indicators of possible variations on the resource availability.

The RKU manages the following events:

**Internal**: related to unexpected changes in the availability. The event may affect or not the Usage Agenda. When the unavailability is produced during an idle period for the resource, the event is logged but no exception is generated. If the unavailability does affect the agenda, an exception is generated and every affected SP agent is notified.

**External**: these are events received through messages coming form SP agents. These events refer to intentions of the SP agent in postponing or expediting the requirement. The RKU agent evaluate the feasibility of accepting the change absorbing the variation with its own safety margins, and accepts or reject the proposition sending a message to the SP agent.

**SP Agents**

SP agents represent a process for the transition from one or more source SKUs to one or more final SKUs. There are three basic transitions: material change (\(\Delta m\)), packing change (\(\Delta p\)) and location change (\(\Delta l\)). Composite transformations are also possible yielding 7 types of transformations.

An SP agent controls the extent the process is executed, detecting variations both in quantity and time. Therefore, its state variable is the process extension, measured in any suitable quantity. For example, a production process can express its extent as the produced quantity. This is equivalent to the verification that the process is being carried out according to plan. In order to support this function, the agent maintains an *Activity Plan*, which can be defined as a list of activities or operations ordered by time. The Activity Plan also tracks the resources assigned for the execution. Every item in the Activity Plan has the following structure:

\[
[\text{Operation ID}, \text{Start Time}, \text{Duration}, \{\text{Bill-of-Materials}\}]
\]

where Bill-of-Materials (BOM) is a list that represents the material input/output structure of the process. Every item in the BOM has the following structure:

\[
[\text{I/O}, \text{SKU}, \text{Quantity}]
\]

The monitored variable of the SP agent is the Current Value for the extension of the process as defined in the state variable.

The variables controlled by the SP agent are the parameters of the Activity Plan: Process Start Time, Process End Time, Input SKUs Quantities and Output SKU Quantities.

The events managed by the SP agent are related with quantity and time variations in the activity they represent. These events can also be classified as internal, external or simulated.

When an internal event is produced, the SP agent starts a negotiation process involving all related SKU and RKU agents. With SKU agents the negotiation is over the Input/Output list and with the RKU agents the subject of negotiation is the Usage Agenda.

An external event can be originated by an RKU agent notifying a change in the availability that may affect the extension of the process or by an SKU that is notifying a change in the Input/Output List.

### 4. Conclusions

Event management is a complex control problem. For any practical application it involves thousands of checkpoints. The affected variables are usually interrelated and these relationships may not be explicit in a global perspective. Our approach based on the decomposition of the global problem into a set of simpler interwoven sub-problems has allowed the causal relationship among state variables to be explicitly defined pair wise and the global interactions described by a network of control points.

This model has allowed to develop an efficient EMS by giving the system some autonomy to define control actions within the safety margins specified in the plan to mitigate the effects of exceptions. The EMS has been modeled as a set of interrelated sub-systems, each of them representing a control point that has a plan and its safety margins defined by the PS.
Due to the model is strongly based in autonomy of the subsystems that is a typical characteristic of software agent technology, the EMS entities have been modeled by autonomous or semi-autonomous agents where the knowledge and abilities are locally distributed and the communication between them is performed through messages. Then, from the agent vision, the proposed model for the EMS is an agency that defines a machine developed to support the event management by means of the cooperation among agents.

As an example, the model has been used to develop an efficient SCEM system to support the management of events during the execution of supply chain operations. We have modeled the SCEM system as a network of inventory control points that are linked among them by supply process control points. The supply processes use resources that also are modeled as resource control points.

The Gaia methodology extended by the approach of Villarreal, et al, (2002) has been a good support to specificity roles and protocols to define de agent model of the SCEM system.

References